A Standard Description of Grids used in Earth System Models

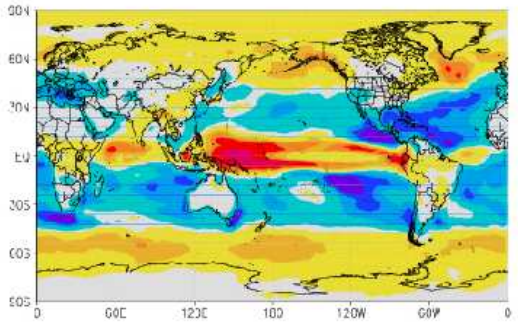
V. Balaji
Princeton University and NOAA/GFDL

GO-ESSP Workshop
Lawrence Livermore National Laboratory
Livermore, CA
20 June 2006

The IPCC AR4 archive at PCMDI

The IPCC data archive at PCMDI is a truly remarkable resource for the comparative study of models. Since it came online in early 2005, it has been a resource for ∼300 scientific papers aimed at providing consensus and uncertainty estimates of climate change, from ∼20 state-of-the-art climate models worldwide.

Model	Modeling Center
BCCR BCM2	Bjerknes Centre for Climate Research
CCCMA CGCM3	Canadian Centre for Climate Modeling & Analysis
CNRM CM3	Centre National de Recherches Meteorologiques
CSIRO MK3	CSIRO Atmospheric Research
GFDL CM2_0	Geophysical Fluid Dynamics Laboratory
GFDL CM2_1	Geophysical Fluid Dynamics Laboratory
GISS AOM	Goddard Institute for Space Studies
GISS EH	Goddard Institute for Space Studies
GISS ER	Goddard Institute for Space Studies
IAP FGOALS1	Institute for Atmospheric Physics
INM CM3	Institute for Numerical Mathematics
IPSL CM4	Institut Pierre Simon Laplace
MIROC HIRES	Center for Climate System Research
MIROC MEDRES	Center for Climate System Research
MIUB ECHO	Meteorological Institute University of Bonn
MPI ECHAM5	Max Planck Institute for Meteorology
MRI CGCM2	Meteorological Research Institute
NCAR CCSM3	National Center for Atmospheric Research
NCAR PCM1	National Center for Atmospheric Research
UKMO HADCM3	Hadley Centre for Climate Prediction



This figure, from Held and Soden (2005), is a composite across the entire IPCC archive.

Computational load at GFDL:

- 5500 model years run.
- Occupied half of available compute cycles at GFDL for half a year (roughly equivalent to 1000 Altix processors).
- 200 Tb internal archive; 40 Tb archived at GFDL data portal; 4 Tb archived at PCMDI data portal.

Rationale for a grid standard

Experience from the international modeling campaigns such as IPCC indicates that there is a wide diversity in the model grids used; and further, it appears that this diversity is only increasing.

However, in the absence of a standard representation of grids, it has been rather difficult to perform comparative analyses of data from disparate model grids.

Modeling centers generally develop some site-specific representation of grids for internal use. As models and model components move toward interoperability, it becomes necessary to develop a common representation of model grids.

We propose here standard metadata for grid description that serves the needs of both coupled models and data analysis.

Grid metadata: requirements

- the standard will describe the grids commonly used in Earth system models from global scale to fine scale, and also with an eye looking forward (toward emerging discrete representations) and sideways (to allied research domains: space weather, geosciences);

- the standard will contain all the information required to enable commonly performed scientific analysis and visualization of data, including differential and integral operations on scalar and vector fields;

- the standard will contain all the information required to perform transformations from one model grid to another, satisfying constraints of conservation and preservation of essential features, as science demands (e.g variance conservation, streamline preservation);

- the standard will make possible the development of shared regridding software, varying from tools deployable as web services to perform on-the-fly regridding from data archives, to routines to be used within coupled models. It will enable, but not mandate, the use of these standard techniques.

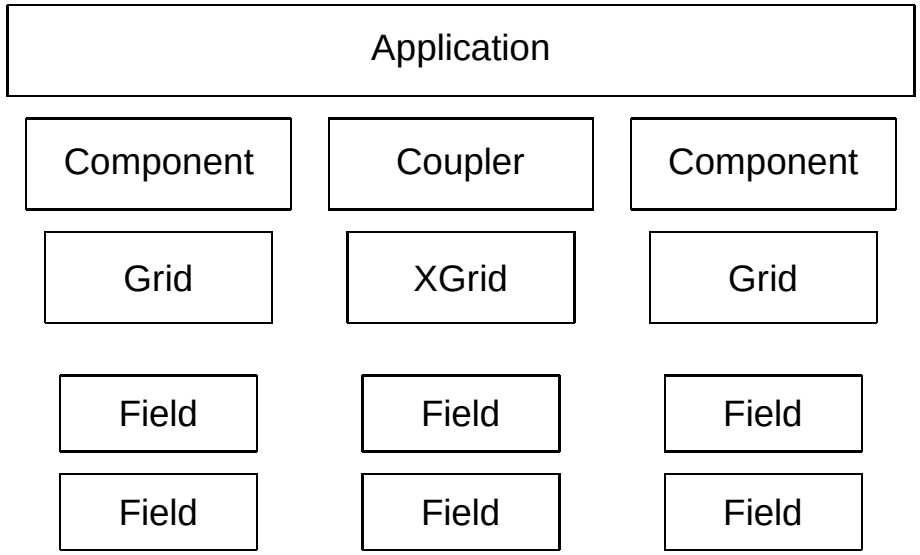
The standard is specifically being proposed for inclusion in the CF standard. The examples here show it in CDL; but it is also available as a schema.

Gridspec within the metadata hierarchy

Application metadata experiment, scenario, institution, contact: currently covered by CF/CMOR.

Component metadata physical and technical description of component and its input and configuration parameters. Currently covered by CMOR, but as free-form text.

Coupler metadata export and import fields, interpolation methods. Currently covered by OASIS4 XML, but not exported to model output. Associated with an XGrid: unstructured grid for fractions and masks. May contain a physical component (e.g surface boundary layer).



Grid metadata geospatial information somewhat covered by CF, but bundled with fields; draft proposal for structural metadata in the works, being negotiated within PRISM, ESMF and GO-ESSP communities, will be proposed as a draft CF standard in 2006.

Field metadata covered by CF/CMOR standard variable name table. Many output fields do not (and should not) have standard names. In general, all metadata categories should allow both standard and bespoke elements.

Grid metadata: what's included

We have chosen two classes of operations that the grid standard must enable: ***vector calculus***, differential and integral operations on scalar and vector fields; and ***conservative regridding***, the transformation of a variable from one grid to another in a manner that preserves chosen moments of its distribution, such as area and volume integrals of 2D and 3D scalar fields. We recognize that higher-order methods that preserve variances or gradients may entail some loss of accuracy. In the case of vector fields, grid transformations that preserve streamlines are required.

To enable vector calculus and conservative regridding, the following aspects of a grid must be included in the specification:

- ***distances*** between gridpoints, to allow differential operations;

- ***angles*** of grid lines with respect to a reference, usually geographic East and North, to enable vector operations. One may also choose to include an ***arc type*** (e.g "great circle"), which specifies families of curves to follow while integrating a grid line along a surface.

- ***areas*** and ***volumes*** for integral operations. This is generally done by defining the boundaries of a grid cell represented by a point value. Below we will also consider fractional areas and volumes in the presence of a ***mask***, which defines the sharing of cell between two or more components.

Geometry and coordinate systems

The underlying ***geometry*** is often a ***sphere*** or thin spherical shell. This may be a problem when geo-referencing to very precise datasets that consider the surface as a geoid. However, more idealized studies may use geometries that simplify the rotational properties of the fluid, such as an f-plane or β-plane, or even simply a cartesian geometry.

Where the actual Earth or planetary system is being modeled, ***geospatial mapping*** or ***geo-referencing*** is used to map model coordinates to standard spatial coordinates, usually ***geographic longitude*** and ***latitude***. Vertical mapping to pre-defined levels (e.g height, depth or pressure) is also often employed as a standardization technique when comparing model outputs to each other, or to observations.

Vertical coordinate

The vertical coordinate can be ***space-based*** (height or depth with respect to a reference surface) or ***mass-based*** (pressure, density, potential temperature). ***Hybrid*** coordinates with a mass-based element are considered to be mass-based.

The ***reference surface*** is a digital elevation map of the planetary surface. This can be a detailed topography or bathymetry digital elevation dataset, or a more idealized one such as the representation of a single simplified mountain or ridge, or none at all. Vertical coordinates requiring a reference surface are referred to as ***terrain-following***. Both space-based (e.g Gal-Chen, ζ) and mass-based (e.g σ) terrain-following coordinates are commonly used.

The rationale for developing this minimal taxonomy to classify vertical coordinates is that translating one class of vertical coordinate into another is generally model- and problem-specific, and should ***not*** be attempted by standard regridding software.

Horizontal coordinate

Horizontal spatial coordinates may be ***polar*** (θ,ϕ) coordinates on the sphere, or ***planar*** (x,y), where the underlying geometry is cartesian, or based on one of several ***projections*** of a sphere onto a plane. Planar coordinates based on a spherical projection define a ***map factor*** allowing a translation of (x,y) to (θ,ϕ).

Curvilinear coordinates may be used in both the polar and planar instances, where the model refers to a pseudo-longitude and latitude, that is then mapped to geographic longitude and latitude by geo-referencing. Examples include the displaced-pole grid and the tripolar grid.

Horizontal coordinates may have the important properties of ***orthogonality*** (when the Y coordinate is normal to the X) and ***uniformity*** (when grid lines in either direction are uniformly spaced). Numerically generated grids may not be able to satisfy both constraints simultaneously. These properties are used as keywords in the gridspec.

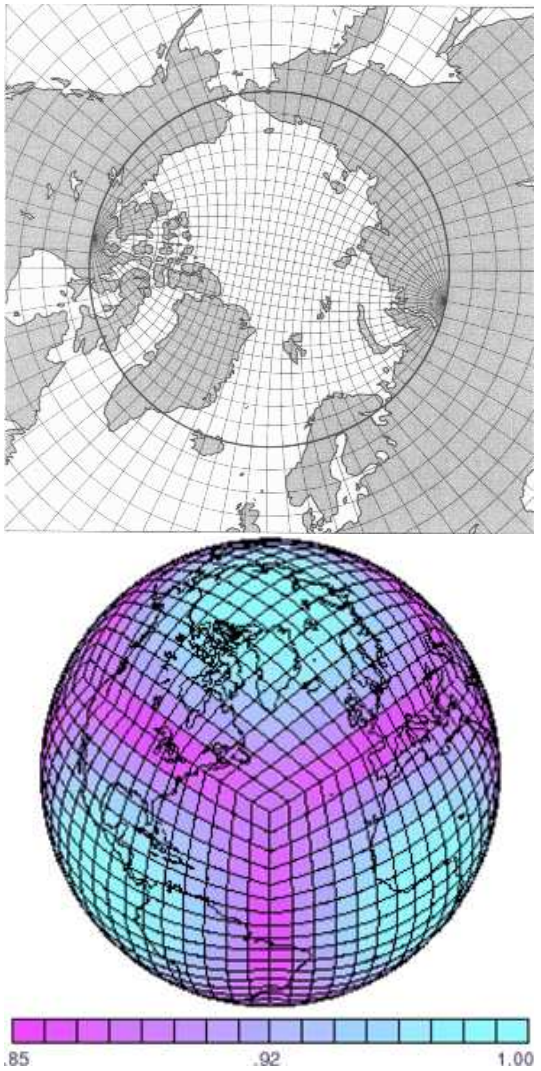
Other coordinate types: spectral, generalized Galerkin methods.

Discretization

The most commonly used discretization in Earth system science is ***logically rectangular***.

A discretization is logically rectangular if the coordinate space (x, y, z) is translated one-to-one to index space `(i,j,k)`. Note that the coordinate space may continue to be physically curvilinear; yet, in index space, grid cells will be rectilinear boxes.

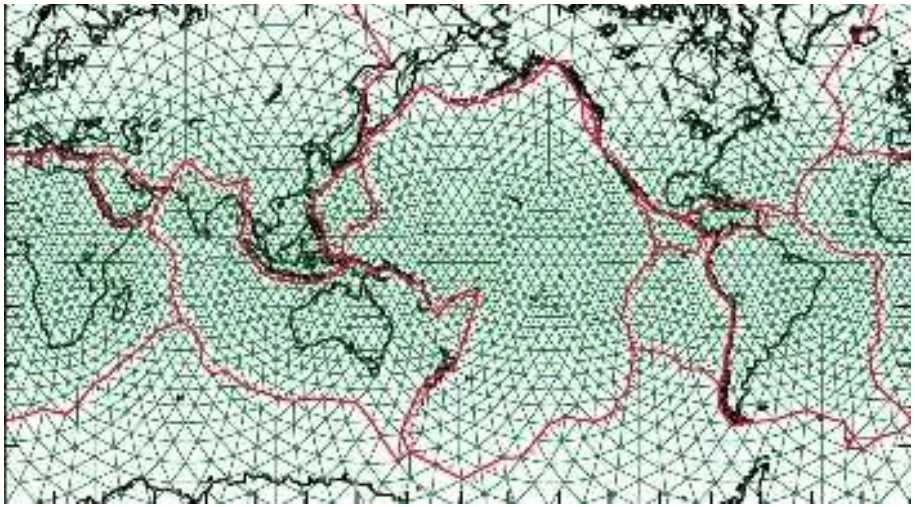
Beyond the simplest logically rectangular grids (e.g lon-lat) we may include more specialized grids such as the tripolar grid (Murray 1996, Griffies et al 2004) and the cubed-sphere grid (Ran cic and Purser 1996).



Triangular discretization

Triangular discretizations are increasingly voguish in the field. A ***structured triangular*** discretization of an icosahedral projection is a popular new approach resulting in a geodesic grid (Majewski et al 2002, Randall et al 2001).

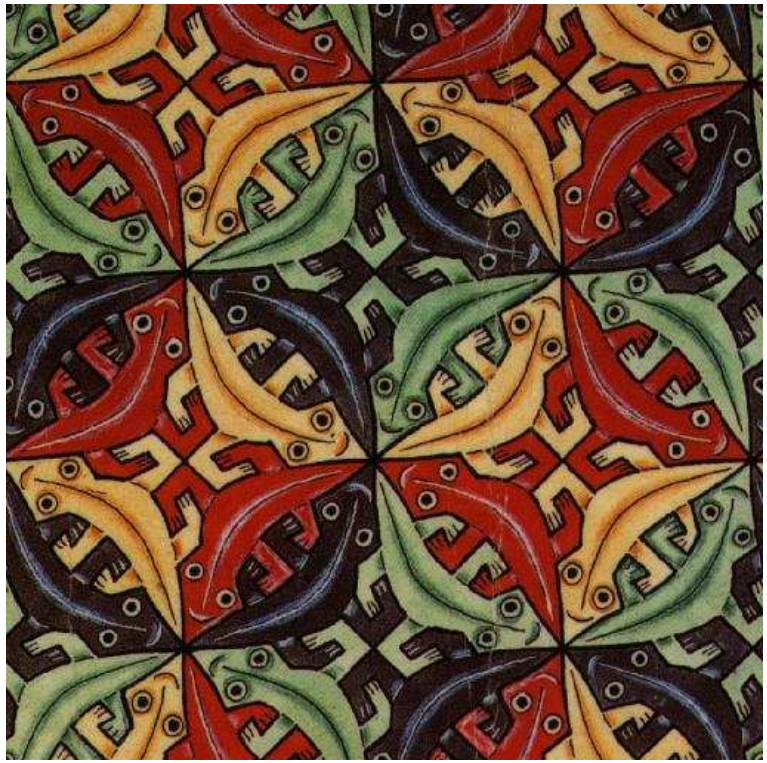
Numerically generated ***unstructured triangular*** discretization is sometimes used, especially over complex terrain.



Grid discretizations: a taxonomy

A reasonably complete taxonomy of grid discretizations for the near- to mid-future in Earth System science would include:

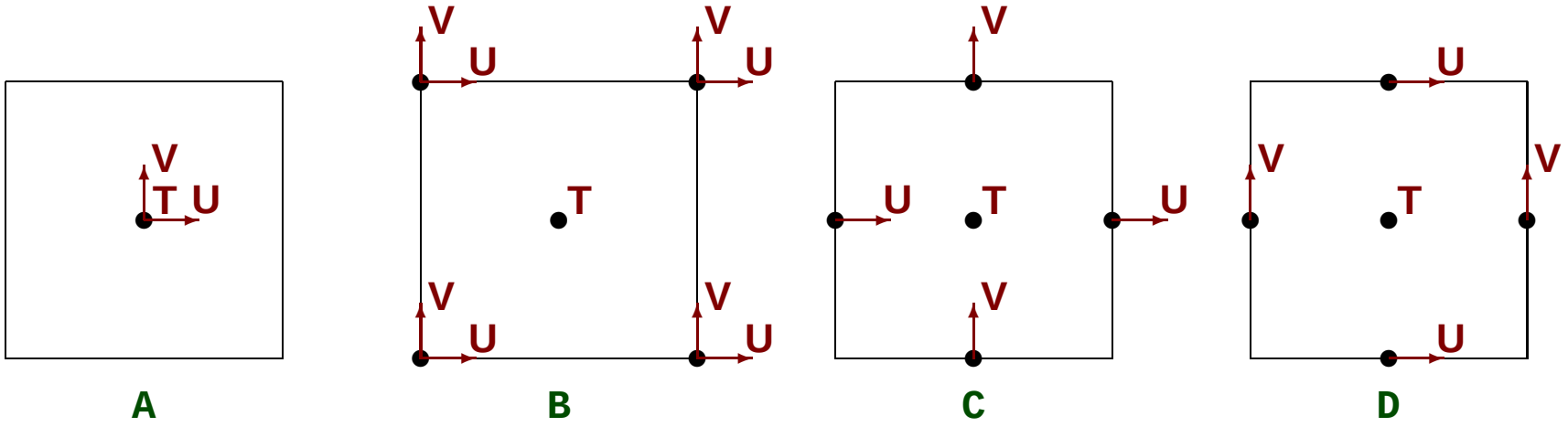
LRG logically rectangular grid.
STG structured triangular grid.
UTG unstructured triangular grid.
UPG unstructured polygonal grid.
PCG pixel-based catchment grids: gridboxes made up of arbitrary collections of contiguous fine-grained pixels, usually used to demarcate ***catchments*** defined by surface elevation isolines.
EGG Escher gecko grid.



While developing a vocabulary and placeholders for all of the above, we shall focus here principally on logically rectangular discretizations. We expect the specification to be extended to other discretization types by the relevant domain experts.

Actual grids may be constructed as a ***grid mosaic*** composed of ***grid tiles***.

Staggering and supergrids

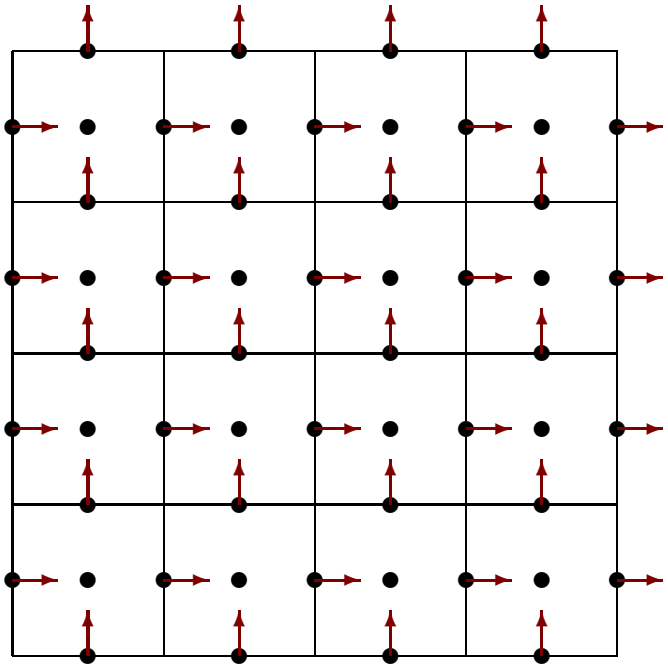


Algorithms place quantities at different locations within a grid cell ("staggering"). This has led to considerable confusion in terminology and design: are the velocity and mass grids to be constructed independently, or as aspects ("subgrids") of a single grid? How do we encode the relationships between the subgrids, which are necessarily fixed and algorithmically essential?

In this specification, we dispense with subgrids, and instead invert the specification: we define a **supergrid**. The ***supergrid*** is an object potentially of higher refinement than the grid that an algorithm will use; but every such grid needed by an application is a subset of the supergrid.

Triangular grid algorithms also use concepts like cell-, face- or vertex-centered quantities, which can be encapsulated within a supergrid.

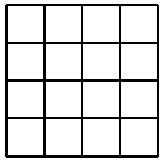
Staggered grid arrays



Shown at left is a supergrid with 9×9 vertices. Interpreted as an Arakawa C-grid, the actual grid contains 4×4 grid cells, with face-centered normal velocities.

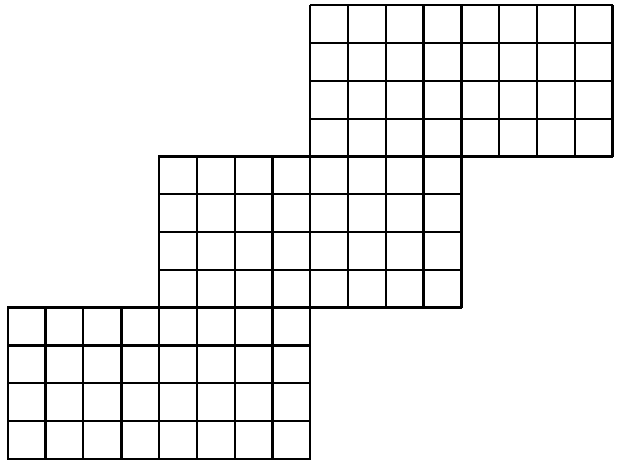
Scalar arrays on this grid are 4×4; if velocity arrays are also 4×4, the staggering is biased in some direction, e.g a northeast-biased staggered grid means that the array value `u(i,j)` represents the point $u_{i+\frac{1}{2}j}$. A symmetric array contains an extra point: e.g a 5×4 array for U. A standard vocabulary for staggered grid arrays is used below.

What is a grid mosaic?

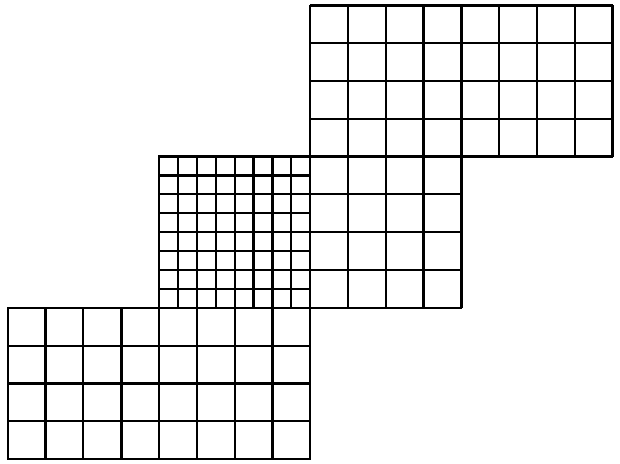


On the left is a basic 4×4 ***tile***; on the right are examples of grids composed of a mosaic of such tiles. The first is a ***continuous grid***, below is a ***refined grid***.

Most current software only supports what we call ***grid tiles*** here. The **`grid mosaic`** extension will allow the development of more complex grids for next-generation models.
First in our (GFDL's) sights is the ***cubic sphere***, primarily targeted at a next-generation finite-volume atmospheric dynamical core, but potentially others as well.
Further developments will include support for irregular tiling (e.g of the ocean surface following coastlines), and for refined, nested and adaptive grids.
Also, regular grids where an irregular decomposition is needed (e.g for a polar filter) can use mosaics to define different decompositions in different regions.



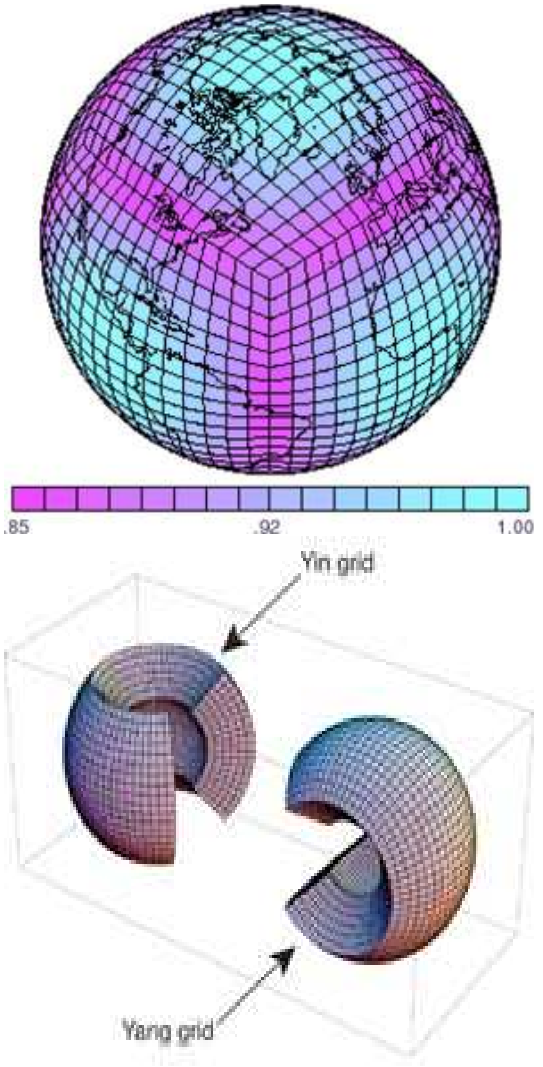
Regular grid mosaic.



Refined grid mosaic.

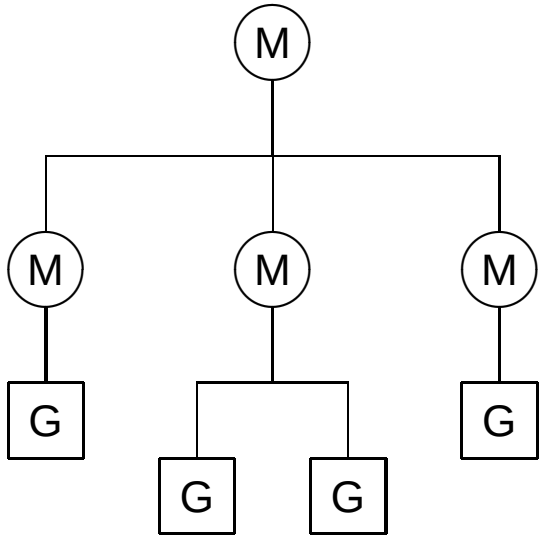
Boundaries and contact regions

Aside from the grid information in the grid tiles, the grid mosaic additionally specifies connections between pairs of tiles in the form of ***contact regions*** between ***pairs*** of grid tiles.
Contact regions can be ***boundaries***, topologically of one dimension less than the grid tiles (i.e, planes between volumes, or lines between planes), or ***overlaps***, topologically equal in dimension to the grid tile. In the cubed-sphere example the contact regions between grid tiles are 1D boundaries: other grids may contain tiles that overlap. In the example of the ***yin-yang*** grid (Kageyama et al 2004) the grid mosaic contains two grid tiles that are each lon-lat grids, with an overlap. The overlap is also specified in terms of a ***contact region*** between pairs of grid tiles. Issues relating to boundaries are described below. Overlaps are described in terms of an exchange grid, more on which below.



Grid mosaic definition

A ***grid mosaic*** is constructed recursively by referring to child mosaics, with the tree terminating in leaves defined by ***grid tiles***.
(There is a very useful analogy to be made between mosaic hierarchies and the component hierarchies we have been talking about in the NMM/ESC context).



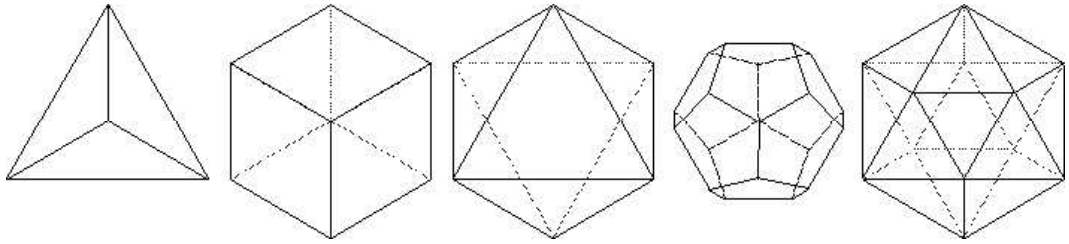
It is not necessarily possible to deduce contact regions by geospatial mapping: there can be applications where geographically collocated regions do ***not*** exchange data, and also where there is implicit contact between non-collocated regions.

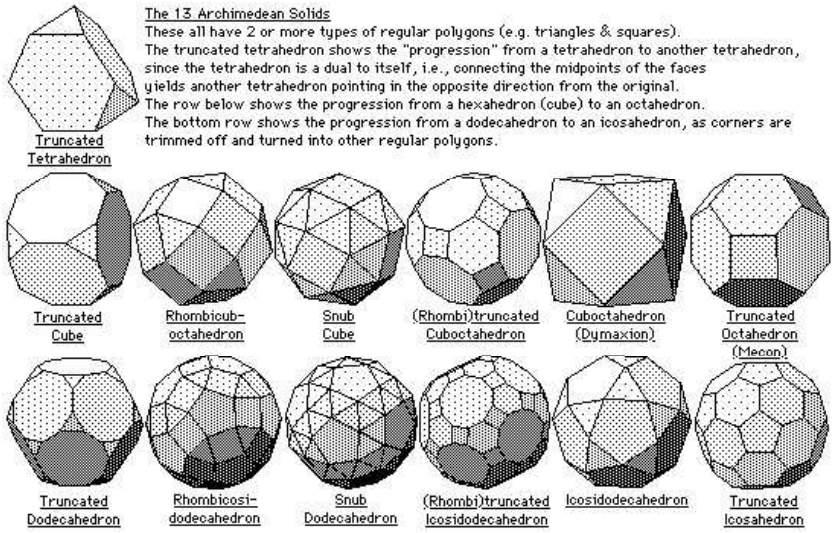
Applications of grid mosaics

The grid mosaic is a powerful abstraction making possible an entire panoply of applications. These include:

- the use of overset grids such as the yin-yang grid;
- the representation of nested grids (e.g Kurihara et al 1990);
- the representation of reduced grids (e.g Rasch 1994). Currently these typically use full arrays and a specification of the "ragged edge". A reduced grid can instead be written as a grid mosaic where each reduction appears as a separate grid tile.
- An entire coupled model application or dataset can be constructed as a hierarchical mosaic. Grid mosaics representing atmosphere, land, ocean components and so on, as well as contact regions between them, all can be represented using this abstraction. This approach is already in use at many modeling centres including GFDL, though not formalized.
- Finally, grid mosaics can be used to overcome performance bottlenecks associated with parallel I/O and very large files. Representing the model grid by a mosaic permits one to save data to multiple files, and the step of ***aggregation*** is deferred. This approach is already used at GFDL to perform distributed I/O from a parallel application, where I/O aggregation is deferred and performed on a separate I/O server sharing a filesystem with the compute server.

UPG mosaics are likely to become more common



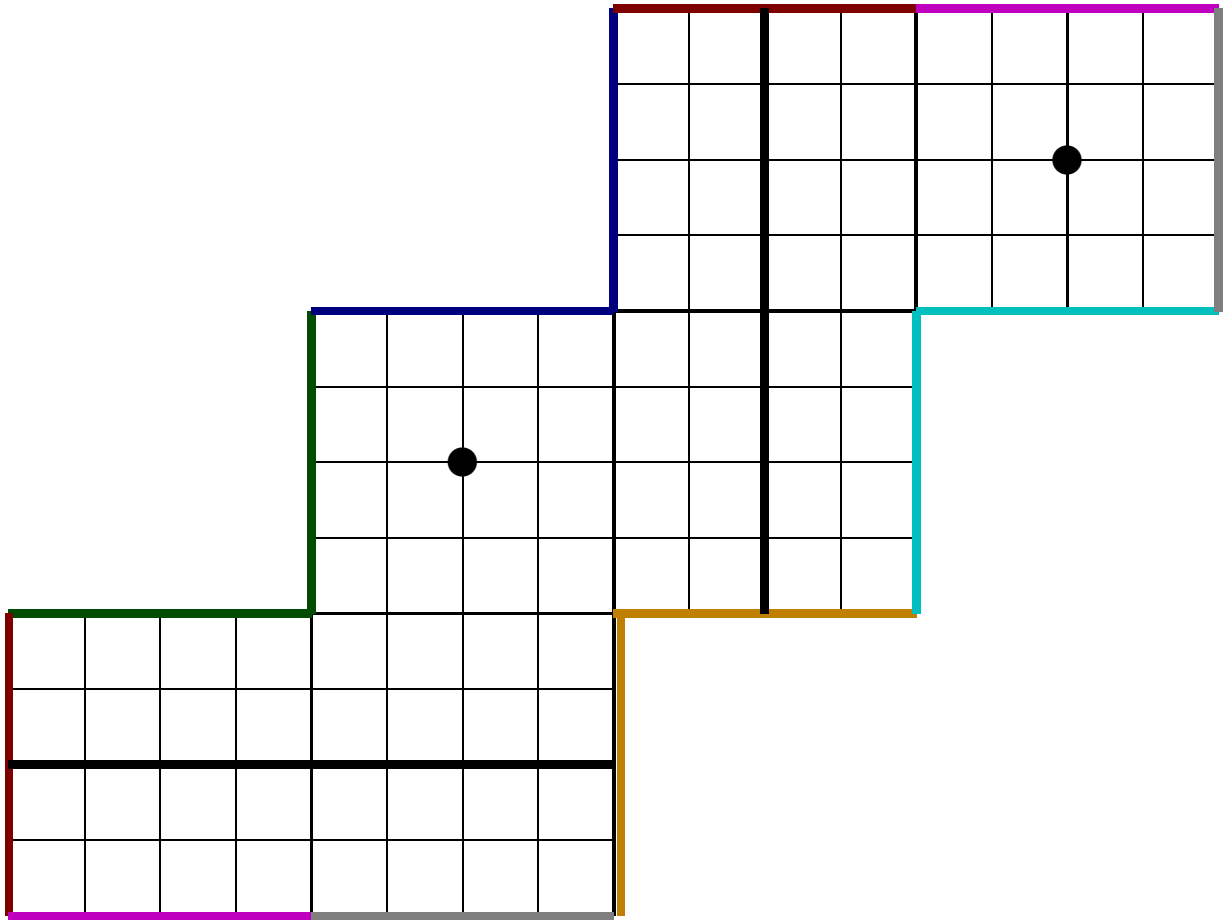


Boundary spec for a cubed sphere

Boundaries for LRG tiles are specified in terms of an ***anchor point*** and an ***orientation***.

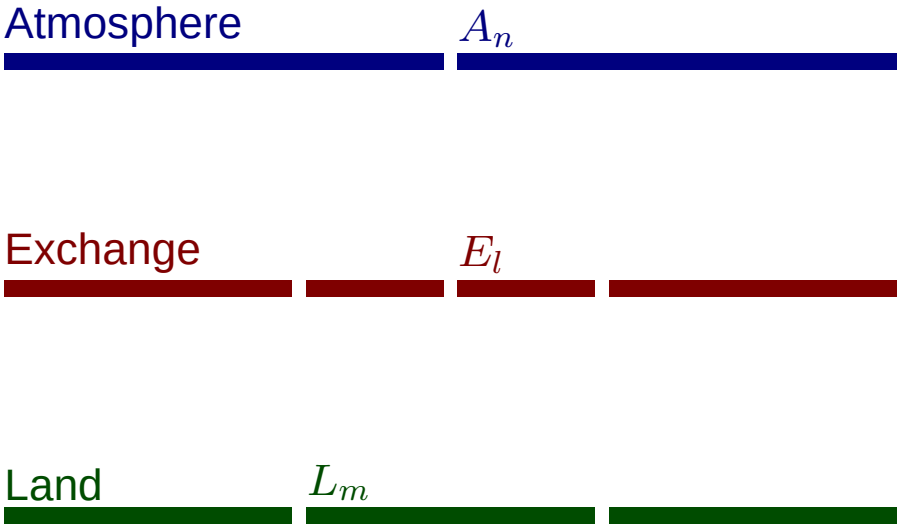
An anchor point is a boundary point that is common to the two grid tiles in contact. When possible, it is specified as integers giving index space locations of the anchor point on the two grid tiles. When there is no common grid point, the anchor point is specified in terms of floating point numbers giving a geographic location.

The ***orientation*** of the boundary specifies the index space direction of the running boundary on each grid tile: the point just to the "west" of **`(5,6)`** is in fact **`(3,4)`**



Overlap contact regions: the exchange grid

- A ***grid*** is defined as a set of ***cells*** created by ***edges*** joining pairs of ***vertices*** defined in a discretization.
- An ***exchange grid*** is the set of cells defined by the union of all the vertices of the two parent grids, and a ***fractional area*** with respect to the parent grid cell.



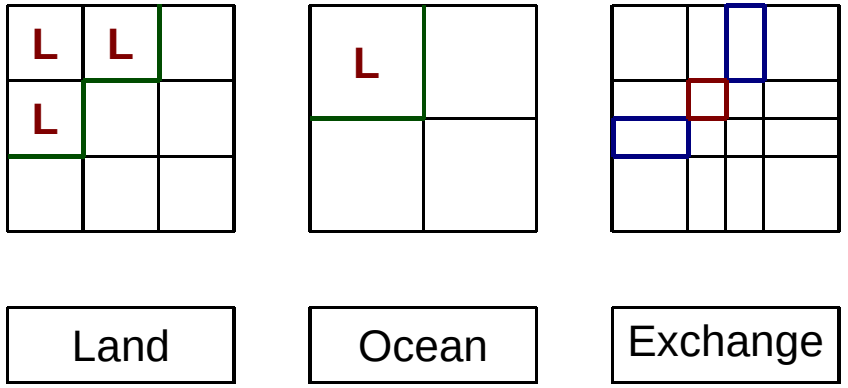
- Exchange: interpolate from source grid using one set of fractional areas; then average onto the target grid using the other set of fractional areas.

- Consistent moment-conserving interpolation and averaging functions of the fractional area may be employed.

Overlap contact regions: masks

Complementary components: in Earth system models, a typical example is that of an ocean and land surface that together tile the area under the atmosphere.

Land-sea mask as discretized on the two grids, with the cells marked **L** belonging to the land. Certain exchange grid cells have ambiguous status: the two blue cells are claimed by both land and ocean, while the orphan red cell is claimed by neither.



Therefore the mask defining the boundary between complementary grids can only be accurately defined on the exchange grid.

In the FMS exchange grid, by convention (and because it is easier) we generally modify the land grid as needed. We add cells to the land grid until there are no orphan “red” cells left on the exchange grid, then get rid of the “blue” cells by ***clipping*** the fractional areas on the land side.

Representing the grid vocabulary in CF

- a standard grid specification dataset (or ***gridspec***) expressed in netCDF or XML. The grid specification is comprehensive and is potentially a very large file. Keywords maybe used in a succinct description from which the complete gridspec is readily reconstructed.

- the current CF spec covers single grid tiles: we have tried to remain close to that spec to preserve legacy data.

- an extended family of CF standard names for grid specification;

- netCDF and CF currently assume that all information is present in a single ***file***, already a flawed assumption (long time series, vector fields). We propose here a mechanism for storing a CF-compliant ***dataset*** in multiple ***files***, and for preserving (or at least verifying) integrity of a multi-file dataset. Fields discretized on a grid mosaic may be held in multiple files.

- Grid metadata is stored below experiment and model metadata. Datasets holding physical variables acquire the gridspec by reference.

- The gridspec is a work in progress, and is designed for extensibility. We expect to see considerable evolution in the near term. It is therefore liberally sprinkled with ***version*** metadata.

CF: LinkSpec

- Links are directed and acyclic: e.g grid mosaic files point to constituent grid tile files, but the “leaf” files do not point back.
- FIle descriptors may be URIs or pathnames. May be absolute or relative to a base address, as in HTML.
- Timestamps and MD5 checksums are stored as attributes.

```
dimensions:
    string = 255;
variables:
    char base(string);
    char external(string);
    char local(string);
base = "http://www.gfdl.noaa.gov/CM2.1/";
    base:standard_name = "link_base_path";
external = "foo.nc";
    external:standard_name = "link_path";
    external:md5_checksum = "g0bbl3dyg00k";
    external:timestamp = "20060509T012800.33Z";
local = "/home/foo/bar.nc";
    local:standard_name = "link_path";
    local:link_spec_version = "0.2";
```

(1)

CF: GridMosaicSpec

```
dimensions:
    nfaces = 6;
    ncontact = 12;
    string = 255;
variables:
    char mosaic(string);
    char gridfaces(nfaces,string);
    char contacts(ncontact,string);
mosaic = "AM2C45L24";
    mosaic:standard_name = "grid_mosaic_spec";
    mosaic:mosaic_spec_version = "0.2";
    mosaic:children = "gridfaces";
    mosaic:contact_regions = "contacts";
    mosaic:grid_descriptor = "C45L24 cubed_sphere";
gridfaces =
    "Face1",
    "Face2",
    "Face3",
    "Face4",
    "Face5",
    "Face6";
contacts =
    "AM2C45L24:Face1::AM2C45L24:Face2",
    ...
    "AM2C45L24:Face5::AM2C45L24:Face6";
```

(2)

CF: GridMosaicSpec

- The grid mosaic spec is identified by a unique string name which qualifies its interior namespace. Its children can be mosaics or grid tiles. Contact regions are specified between pairs of grid ***tiles*** only, using the fully qualified grid tile spec ***mosaic:mosaic:...:tile***.

- The ***grid_descriptor*** is an optional text description of the grid that uses commonly used terminology, but may not in general be a sufficient description of the field (many grids are numerically generated, and do not admit of a succinct description). Examples of grid descriptors include:
  - **`spectral_gaussian_grid`**
  - **`regular_lon_lat_grid`**
  - **`reduced_gaussian_grid`**
  - **`displaced_pole_grid`** (different from a ***rotated pole grid***: any grid could have a rotated north pole);
  - **`tripolar_grid`**
  - **`cubed_sphere_grid`**
  - **`icosahedral_geodesic_grid`**
  - **`yin_yang_grid`**

CF: GridTileSpec

```
dimensions:
    string = 255;
    nx = 90;
    ny = 90;
    nxv = 91;
    nyv = 91;
    nz = 24;
variables:
    char tile(string);
    double area(ny,nx);
        area:standard_name = "grid_cell_area";
        area:units = "m^2";
    double dx(ny+1,nx);
        dx:standard_name = "grid_edge_x_distance";
        dx:units = "metres";
    double angle_dx(ny+1,nx);
        angle_dx:standard_name =
            "grid_edge_x_angle_WRT_geographic_east";
        units = "radians";
    char arcx(string);
        arcx:standard_name = "grid_edge_x_arc_type";
        arcx:north_pole = "0.0 90.0";
    double zeta(nz);
```

(3)

CF: GridTileSpec

```
arcx = "small_circle";
tile = "Face1";
    tile:standard_name = "grid_tile_spec";
    tile:tile_spec_version = "0.2";
    tile:geometry:  "spherical";
    tile:north_pole:  "0.0 90.0";
    tile:projection:  "cube_gnomonic";
    tile:discretization = "logically_rectangular";
    tile:conformal = "true";
```

(4)

CF: GridTileSpec georeferencing

```
variables:
   float geolon(ny+1,nx+1);
      geolon:standard_name = "geographic_longitude";
   float geolat(ny+1,nx+1);
      geolat:standard_name = "geographic_latitude";
   float x_vertex(ny+1,nx+1);
      x_vertex:standard_name = "grid_longitude";
      x_vertex:geospatial_coordinates = "geolon geolat";
   float y_vertex(ny+1,nx+1);
      y_vertex:standard_name = "grid_latitude";
      y_vertex:geospatial_coordinates = "geolon geolat";
```

(5)

The vertical geo-mapping is expressed by reference to “standard levels”.

CF: GridContactSpec for a boundary

```
dimensions:
    string = 255;
variables:
    int anchor(2,2);
        standard_name =
            "anchor_point_shared_between_tiles";
    char orient(string);
        orient:standard_name =
            "orientation_of_shared_boundary";
    char contact(string);
        contact:standard_name = "grid_contact_spec";
        contact:contact_spec_version = "0.2";
        contact:contact_type = "boundary";
        contact:alignment = "true";
        contact:refinement = "none";
        contact:anchor_point = "anchor";
        contact:orientation = "orient";
contact = "AM2C45L24:Face1::AM2C45L24:Face2";
orient = "Y:Y";
anchor = 90, 1, 1, 1;
```

(6)

Cyclicity and the tripolar fold (**`orient = "X:-X"`**) can both be expressed as a boundary of a grid tile with itself.

CF: GridContactSpec for an overlap

```
dimensions:
    string = 255;
    ncells = 1476;
variables:
    double frac_area(2,ncells);
        standard_name =
            "fractional_area_of_exchange_grid_cell";
    int tile1_cell(2,ncells);
        standard_name="parent_cell_indices";
    int tile2_cell(2,ncells);
        standard_name="parent_cell_indices";
    char contact(string);
        contact:standard_name = "grid_contact_spec";
        contact:contact_spec_version = "0.2";
        contact:contact_type = "exchange";
        contact:fractional_area_field = "frac_area";
        contact:parent1_cell = "tile1_cell";
        contact:parent2_cell = "tile2_cell";
contact = "CM2:LM2::AM2C45L24:Face2";
```

(7)

Data files using the gridspec

```
dimensions:
    nx = 46;
    ny = 45;
variables:
    int nx_u(nx);
    int ny_u(ny);
    float u(ny,nx);
        u:standard_name = "grid_eastward_velocity";
        u:staggering = "c_grid_symmetric";
        u:coordinate_indices = "nx_u ny_u";
GLOBAL ATTRIBUTES:
    gridspec = "/foo/gridspec.nc";
nx_u = 1,3,5,...
ny_u = 2,4,6,...
```

(8)

Variables on a single grid tile can follow CF-1.0, with no changes.

The ***staggering*** field expresses what is implicit in the values of **nx_u** and **ny_u**. Possible values of **staggering** include**c_grid_symmetric**, **c_grid_ne** and so on.

Using this information, it is possible to perform correct transformations, such as combining this field with a V velocity from another file, transforming to an A-grid, and then rotating to geographic coordinates.

Summary: an extended gridspec for CF

- We propose a new grid specification standard for CF. It can be expressed in netCDF-3, 4, or XML.

- The specification of a grid tile is consistent with the current CF gridspec, but extends it by defining the supergrid and staggering. Current netCDF-3 data files need not change but for the addition of a few attributes.

- The definition of a grid mosaic is new. Among other things, the mosaic specification can help widen the parallel I/O and filesize bottlenecks.

- The grid specification is maintained separately from the dataset, which links to it. Integrity of linkages between files is maintained by a LinkSpec.

- If the gridspec file is standardized, it can be used for model input as well as output. For coupled or nested models, this file also contains the necessary data to relate component grid mosaics.

What's needed next

- prototype and test across more than one institution;
- CF to agree to an extended standard for gridded datasets;
- PRISM/ESMF to agree to produce compliant data;
- Tools to become capable of applying standard and bespoke regridding techniques.
- When submitting the draft gridspec to CF, GFDL will supply tools to create gridspec files, and visualize fields based on the gridspec (end of summer 2006).

Questions?